BatLab Basic Project Kit – Blink an LED

HOW IT WORKS

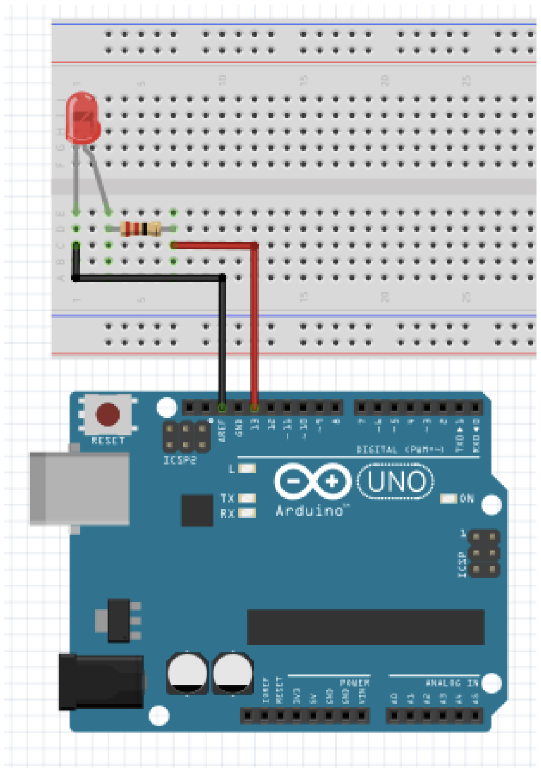
LEDs are small, powerful lights that are used in many different applications. They are polarized which means that the current can only go one way through them. The longer leg of the LED is the positive terminal, or anode; the shorter leg is the negative terminal, or cathode. For the LED to light up, it must be connected in a circuit with an appropriate voltage source (here, the Arduino) and a resistor. The resistor is important! If there is no resistor, there will be too much current and the LED will burn out quickly. If your LED is not turning on, be sure that that it's oriented properly (you can't hurt it by putting it in backwards; it just won't work).

PARTS

- Arduino Uno, breadboard, and jumper wires
- LED
- 220Ω resistor (red marking)

CIRCUIT

In this diagram, the longer leg of the LED is to the right.



WIRING

Note that one pin on the LED is longer than the others. This is the anode, or positive terminal. Be sure that it is connected to the resistor as shown in the diagram.

CODE

```
/*
BatLab Basic Project Kit

BLINKING AN LED

Hardware connections:


Connect the positive side of your LED (longer leg) to Arduino digital
pin 13 (or another digital pin, don't forget to change the code to
match).

Connect the negative side of your LED (shorter leg) to a 220 Ohm
resistor (red-red-black-black). Connect the other side of the resistor
to ground.

[We always use resistors between the Arduino and and LEDs to keep the
LEDs from burning out due to too much current.]

This sketch was based on Circuit #1 from Sparkfun:
https://learn.sparkfun.com/

Visit http://www.arduino.cc to learn about the Arduino.


*/



// Welcome to Arduino!

// If you're brand-new to this, there will be some new things to
// learn, but we'll jump right in and explain things as we go.

// The Arduino is a tiny computer that runs programs called
// "sketches". These are text files written using instructions
// the computer understances. You're reading a sketch right now.

// Sketches have computer code in them, but also (hopefully)
// "comments" that explain what the code does. Comments and code
// will have different colors in the editor so you can tell them
// apart.

// This is a comment - anything on a line after "//" is ignored
// by the computer.

/* This is also a comment - this one can be multi-line, but it
must start and end with these characters */

// A "function" is a named block of code, that performs a specific,
// well, function. Many useful functions are already built-in to
// the Arduino; others you'll name and write yourself for your
// own purposes.

// All Arduino sketches MUST have two specific functions, named
// "setup()" and "loop()". The Arduino runs these functions
// automatically when it starts up or if you press the reset
// button. You'll typically fill these function "shells" with your
// own code. Let's get started!


// The setup() function runs once when the sketch starts.
// You'll use it for things you need to do first, or only once:


void setup()
{
  // The Arduino has 13 digital input/output pins. These pins
  // can be configured as either inputs or outputs. We set this
  // up with a built-in function called pinMode().

  // The pinMode() function takes two values, which you type in
  // the parenthesis after the function name. The first value is
  // a pin number, the second value is the word INPUT or OUTPUT.

  // Here we'll set up pin 13 (the one connected to a LED) to be
  // an output. We're doing this because we need to send voltage
  // "out" of the Arduino to the LED.

  pinMode(13, OUTPUT);

  // By the way, the Arduino offers many useful built-in functions
  // like this one. You can find information on all of them at the
  // Arduino website: http://arduino.cc/en/Reference
}


// After setup() finishes, the loop() function runs over and over
// again, forever (or until you turn off or reset the Arduino).
// This is usually where the bulk of your program lives:


void loop()
```

```
{
  // The 13 digital pins on your Arduino are great at inputting
  // and outputting on/off, or "digital" signals. These signals
  // will always be either 5 Volts (which we call "HIGH"), or
  // 0 Volts (which we call "LOW").

  // Because we have an LED connected to pin 13, if we make that
  // output HIGH, the LED will get voltage and light up. If we make
  // that output LOW, the LED will have no voltage and turn off.

  // digitalWrite() is the built-in function we use to make an
  // output pin HIGH or LOW. It takes two values; a pin number,
  // followed by the word HIGH or LOW:

  digitalWrite(13, HIGH);   // Turn on the LED

  // delay() is a function that pauses for a given amount of time.
  // It takes one value, the amount of time to wait, measured in
  // milliseconds. There are 1000 milliseconds in a second, so if
  // you delay(1000), it will pause for exactly one second:

  delay(1000);              // Wait for one second

  digitalWrite(13, LOW);    // Turn off the LED

  delay(1000);              // Wait for one second

  // All together, the above code turns the LED on, waits one
  // second, turns it off, and waits another second.

  // When the computer gets to the end of the loop() function,
  // it starts loop() over again. So this program will continue
  // blinking the LED on and off!

  // Try changing the 1000 in the above delay() functions to
  // different numbers and see how it affects the timing. Smaller
  // values will make the loop run faster. (Why?)
}
```